From Garage-Band to World Tour: Technical, Security, and Scalability Challenges of a Web-Based Program Management Tool from Workgroup-Level to Enterprise-Class in 24 Months

Track: Information Superiority/Information Operations

Authors: **Helen M. Rico**
Air Force Research Laboratory/ Information Directorate
AFRL/IFGA
525 Brooks Road
Rome, New York 13441
315-330-3432 (phone)
315-330-1995 (fax)
Helen.Rico@rl.af.mil

Fred Hall
Air Force Research Laboratory/ Information Directorate
AFRL/IFGA
525 Brooks Road
Rome, New York 13441
315-330-2306(phone)
315-330-1995 (fax)
Fred.Hall@rl.af.mil

Michael J. Maciolek II
Northrop Grumman IT
525 Brooks Road
Rome, New York 13441
315-330-1459 (phone)
315-330-1995 (fax)
Michael.Maciolek@rl.af.mil

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **JUN 2004**	2. REPORT TYPE	3. DATES COVERED **00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE **From Garage-Band to World Tour: Technical, Security, and Scalability Challenges of a Web-Based Program Management Tool from Workgroup-Level to Enterprise-Class in 24 Months**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Air Force Research Laboratory,Information Directorate (AFRL/IFGA),525 Brooks Road,Rome,NY,13441**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**45**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

From Garage-Band to World Tour: Technical, Security, and Scalability Challenges of a Web-Based Program Management Tool from Workgroup-Level to Enterprise-Class in 24 Months

Helen M. Rico
Air Force Research Laboratory/ Information Directorate
AFRL/IFGA
525 Brooks Road
Rome, New York 13441
315-330-3432 (phone)
315-330-1995 (fax)
Helen.Rico@rl.af.mil

Fred Hall
Air Force Research Laboratory/ Information Directorate
AFRL/IFGA
525 Brooks Road
Rome, New York 13441
315-330-2306(phone)
315-330-1995 (fax)
Fred.Hall@rl.af.mil

Michael J. Maciolek II
Northrop Grumman IT
525 Brooks Road
Rome, New York 13441
315-330-1459 (phone)
315-330-1995 (fax)
Michael.Maciolek@rl.af.mil

Abstract

The Air Force Research Laboratory (AFRL) has a web-based application used for program management which provides its scientists and engineers a clearer, more rapid picture of their contractual and in-house R&D efforts' financial and technical status, by allowing the contractor to enter information directly into the tool. This web application, called Jiffy, began as a tool developed by two engineers in AFRL's Information Directorate (IF) using ASP pages talking to a Microsoft Access database and was initially used by a handful of people. Senior management saw Jiffy as a tool that would benefit all of IF's engineers and scientists (approximately 450 people). Jiffy became recognized by AFRL as a best-practice and an effort was started to scale Jiffy up for use by all Air Force Research Laboratory engineers (approximately 3000 users). In this paper, the authors will describe the issues and solutions in migrating the application from an Access database to an Oracle database (and the technical architecture used), how the security of the application was improved, and how the application performance was enhanced to allow the application to scale up from a handful of users to thousands of users.

Architectural Basics

The Jiffy application is web-based and can be accessed via any web browser capable of 128-bit (HTTPS) communication. The Jiffy architecture consists of two main parts; the *web server* which handles user input and graphical display (along with some program logic), and the *database server* which houses the information gathered from various AF standard systems and electronic copies of documents related to specific research programs. A logical diagram of the Jiffy system is shown in Figure 1.

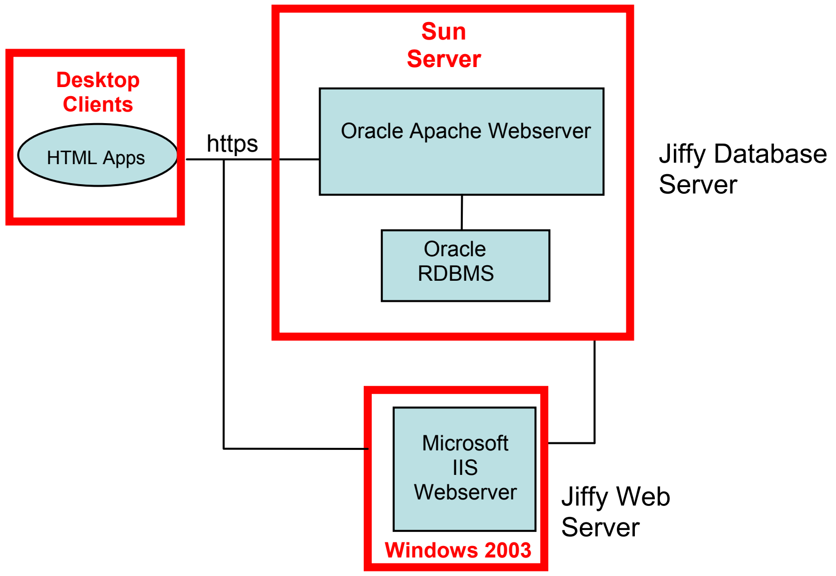


FIGURE 1. Logical Representation of the Jiffy Architecture

The two portions of the Jiffy system; web server and database server, can be housed on the same physical hardware or can be hosted on separate hardware platforms. There are advantages and disadvantages to each physical implementation that will become apparent in the following sections describing the evolution of the architecture and security and performance of the system.

Evolution of the Architecture

The Jiffy architecture has undergone major transitions over its short (approximately 2 ½ year) life span. The reasons can be better understood when you think about the fact that the application went from being used in a work-group environment by a handful of users, to a department-wide application with a few hundred users, to an enterprise-class application with a couple thousand geographically-dispersed users.

When originally stood up as a work-group application, the Jiffy web server and database server both resided on the same Windows-based computer (see Figure 2). The Jiffy ASP application interfaced to a Microsoft Access database. Documents related to research programs were stored in the Windows file system and information from AF standard systems was stored in the database. Microsoft Access is not known for its scalability and robust security in an enterprise environment, but for that period of time when network information assurance was not as critical as today, and given the small number of users, it was sufficient. It was also the easiest database for the two engineers and one programmer who developed the first version of Jiffy to learn and use.

When first put into operation, Jiffy was accessed not only by the workgroup of AF engineers in the .mil domain, but also by their contractors in the .com and .edu domains. Because external connections to workgroup web servers were not allowed through the firewall, the Jiffy hardware had to be placed in an extranet location outside the protection of the base firewall as shown in Figure 2. This created potential security implications which will be discussed in a later section of this paper. However, having the entire system housed on one physical machine, coupled with the small user population, allowed for good application performance. In addition, only one machine had to be administered (patched, updated, etc) keeping operations and maintenance costs low.

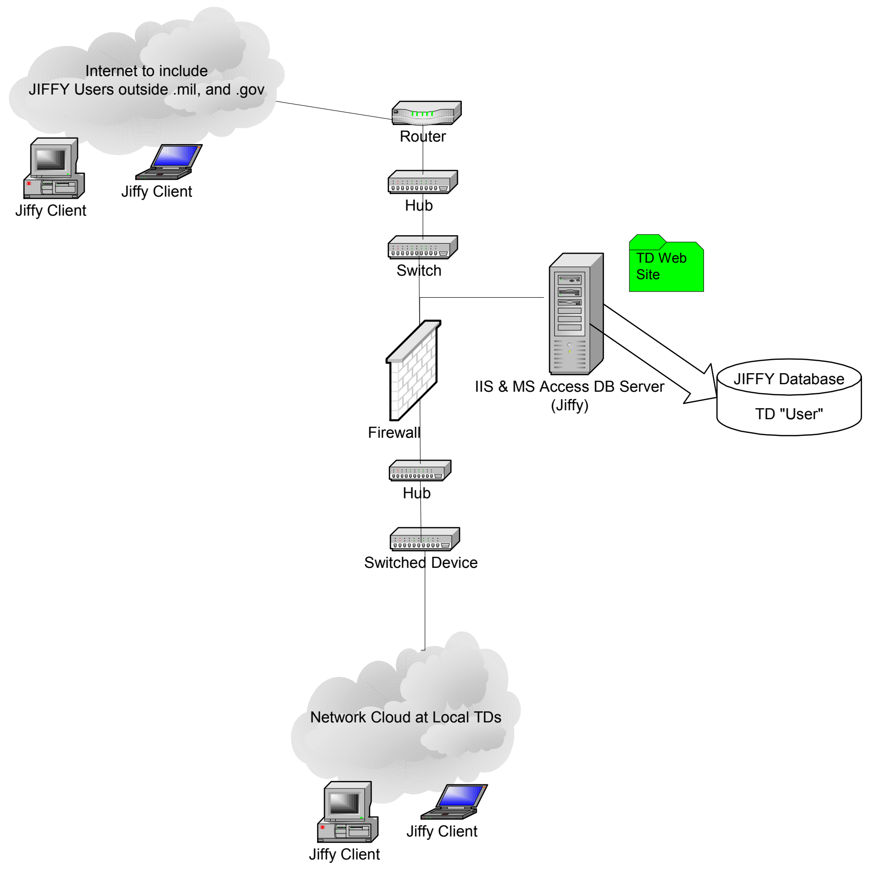


FIGURE 2. Jiffy Workgroup Architecture

As the usefulness of the Jiffy application became apparent to more personnel within the Information Directorate, the IF Director requested that the Jiffy application be made robust to support the entire Directorate (a couple hundred users) along with an increase in the security of the entire application architecture – all within nine months. A team was assembled to make this happen. The Jiffy development team was increased from two engineers and a programmer to two engineers, four programmers and two co-program managers. In order to provide better support to the larger user base, a full-time application support person (i.e. helpdesk) was hired.

Since all of the other IF Directorate-wide applications use Oracle databases on Sun Solaris-based hardware, it was decided to migrate the Jiffy Access database to Oracle. This meant the Jiffy application now required two hardware platforms; a Windows server for the web portion, and a Sun computer for the Oracle database. For better security it was decided to move the entire application architecture inside the base firewall as shown in Figure 3. Firewall rules would be established to allow necessary HTTPS access to the Jiffy application for non-.mil domain users. This new architecture increased the security of the application, but initially adversely affected performance (as will be discussed later in this paper). However, once steps were taken to optimize the application, performance improved dramatically, and this architecture could easily support a few hundred users.

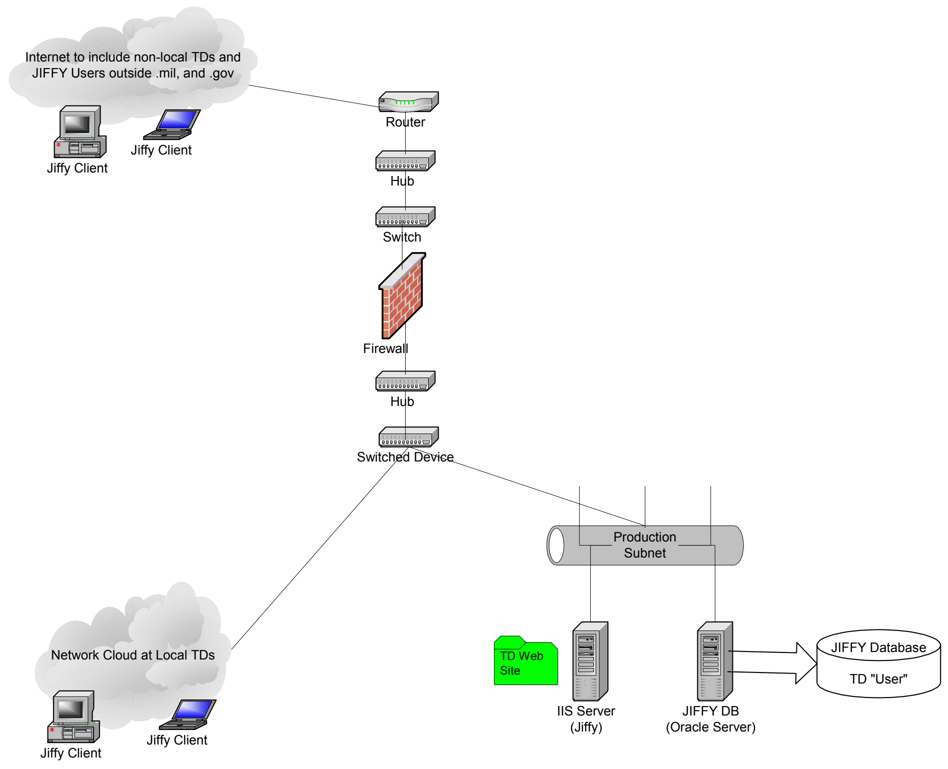


FIGURE 3. Jiffy Directorate-wide Architecture

During the time the IF Directorate was deploying Jiffy, the Commander of AFRL was looking for various applications that could be used across the entire laboratory to establish a common corporate toolset and better align the Lab's internal business processes.

Jiffy was chosen to become the standard program management tool across all nine technology directorates of AFRL. Therefore, the application had to scale up for use by approximately 3,000 engineers deployed at geographically-dispersed sites across the CONUS. The Commander wanted initial deployment of the application lab-wide in 11 months.

To accomplish this new tasking, a program office was formed with Government oversight into the areas of development, architecture, security, test, etc. The team was also bolstered with three additional programmers, two software testers, a QA person, and a part-time software security person. During the course of development, application performance issues arose which required bringing in a paid-consultant for recurring code reviews.

For AFRL-wide deployment, the Jiffy architecture was moved from the IF Directorate (Rome, NY) and hosted at AFRL headquarters at Wright-Patterson AFB, OH. New server hardware was procured based on the projected user load of a few thousand people. As shown in Figure 4, to increase security the web server was placed in a demilitarized zone (DMZ) - essentially attached to the firewall with its own special rule-set to allow necessary access. The diagram shows a somewhat simplified view - in actuality there are three web servers, each hosting the Jiffy application for three of AFRL's technical directorates to balance the user load. These three web servers all connect to a single Sun Solaris database server on the back end. In this deployment of Jiffy, the electronic copies of documents have been moved from the Windows file system to the Oracle database for improved security as will be explained later in this paper. This architecture has proved to be superb at handling the current user load, and application response times are excellent even for users located at AFRL locations in California or Florida.

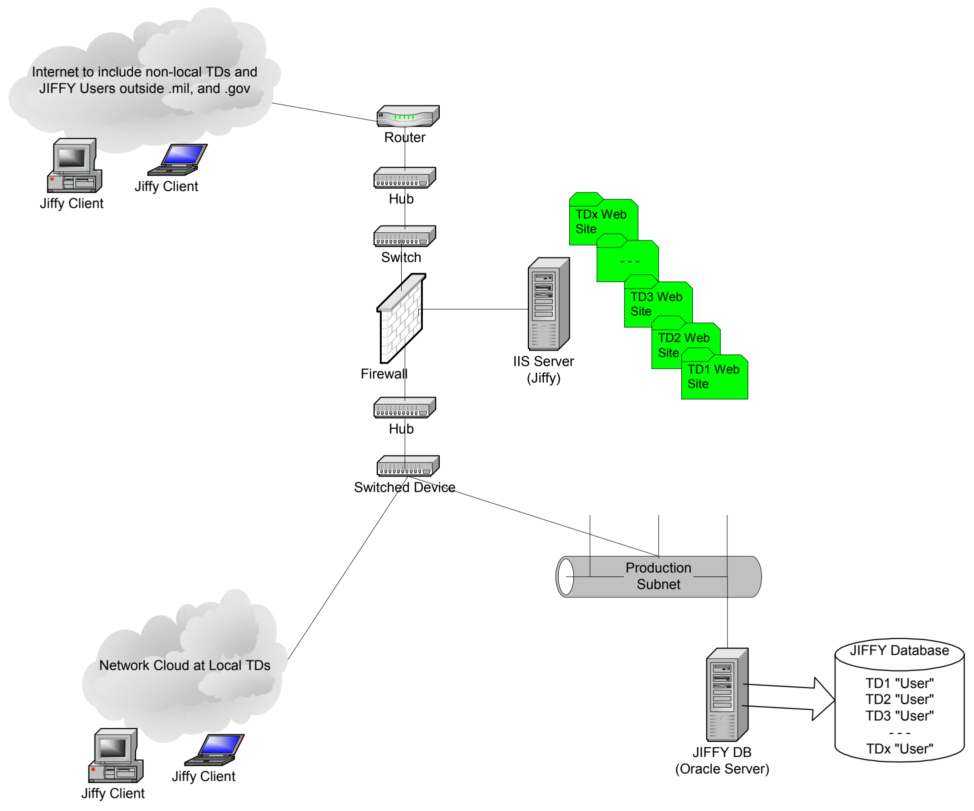


FIGURE 4. Jiffy Enterprise Architecture

Security Considerations

The distributed nature of clients in web-based applications and the textual nature of the web pages themselves forced the Jiffy team to address a wide range of security issues when delivering sensitive information to users. The security aspects of Jiffy can be grouped into several areas including server access, application access, user permissions, and protection from hackers.

Server Access for .com/.edu Users

One of the advantages of the Jiffy application is allowing the contractors performing the technical work on a program to log in and update their relevant financial and technical status information on a regular basis, thus unburdening the engineer from this task and capturing the information closer to the source.

The security issues arise in that most of the contractors reside in the .com or .edu domains, while the Jiffy architecture is in the .mil domain. Today's emphasis on network security, along with AF policy, has to be balanced against the need of users from any internet domain to be able to access the system.

As security policies and needs have evolved, so has the physical Jiffy architecture as shown in the previous sections. In its initial workgroup incarnation, Jiffy was deployed outside the base firewall, and all sensitive financial and technical information was stored on the same machine as the application. To increase the security posture when deployed Directorate-wide, the Jiffy servers were moved inside the base firewall with non-essential services disabled and special firewall rules put in place. This was still a trade-off in that it allowed access by outside domain users to machines behind the base firewall. In its current configuration for enterprise-wide use, the Jiffy web server has been moved to its own leg off the base firewall with special access rules. The database server stays behind the firewall and the only machine allowed access to it is the Jiffy web server. Additionally, electronic document storage, which was once part of the Jiffy web server file system, is now done in the database (on the database server) making the documents much less accessible to would-be hackers.

The team believes this current architecture balances the need for user access against the possible security threats that may exist.

User Agreement for Application Access

The process for authorizing and activating accounts into the JIFFY application is a multi-phased approach. The driving force for this process is because two-thirds of the users (contractors) will never be physically seen by JIFFY Administrative personnel. Therefore, the user account generation process uses a trusted-agent approach.[1] To become a Jiffy user, you must be nominated by a current Jiffy user. All users are required to be citizens of the United States of America or hold a valid Green Card. Approval for Foreign National Access must follow Table 3.1 "Approving Authority for Foreign National Access" as provided in AFI33-202.

User Permissions

Once user access is granted to the application, the role-based permission system comes into play. Every user that successfully logs into Jiffy is provided with a custom user interface that corresponds with functionality based on that user's role. Only features expressly permitted for a user of their role will be present in the navigational menus of Jiffy. The Jiffy developers also implemented an additional permission-based mechanism for the data itself based on row-level access in the database. Users will only get data returned to them that is permitted to them based on a combination of their user ID and role. This double layer of protection ensures that users will only ever get access to pieces of the application they should, and even if they were to navigate somewhere outside of their intended scope, they will still be limited to only the data they have permission to access.

Even if users operate only through allowed sections of the application, they may still wish to compromise sensitive data through various hacking techniques. Jiffy has several protection mechanisms in place to prevent such hacking and to help diagnose attacks after the fact. Jiffy was written to minimize exposure to SQL injection hacks, anonymous file system access, undesired execute privileges, and URL hijacking. Jiffy has both client-side and server-side code to check for and disarm such attacks.

Jiffy also has an effective traceability system for users. This system tracks a user's path through the system and logs important actions they perform and the data they operate on. This, along with system level web-server logs, allows system administrators to isolate the cause of suspect data changes and to quickly reproduce the actions taken during such a compromise. Jiffy's traceability system is also very useful for day-to-day helpdesk level support and debugging.

Scalability

Application performance can be measured in many ways based on criteria established for that specific application, as well as by measuring generally accepted performance metrics. The better the application performs, as measured by comparing response time against concurrent users, the more scalable the application is.

The two main performance measures used to analyze web-based application performance are *response time* and number of *concurrent users*. Response time is defined as the time it takes for the web page to be completely returned to the client's browser from the server once requested. Another way to consider response time is to measure the average number of requests per second that can be handled by the server. The faster the server responses are, the faster it can serve up additional requests. The concurrent user metric refers to the number of simultaneous users that can access and exercise the application at the same time with acceptable server response times. The Jiffy team used automated reliability and load testing tools to help take measurements. You can see in Figure 5 below that Jiffy's original configuration and software allowed for a maximum of 10 concurrent users. The current enterprise level Jiffy version can easily accommodate 100 concurrent users driving maximum load to the web server. Jiffy can now handle, at that maximum load, about 5 times the number of requests as the original version of Jiffy. (see Figure 5) Under normal load it can handle many more requests per second.

The workgroup version of Jiffy did not suffer from systemic server response time issues. This was an unintended benefit of having the database on the same physical platform as the web server and the homogeneous software architecture (all Microsoft). The normal cause of response time problems stems from poor coding practices that lead to bottlenecks under concurrent usage. As shown in Figure 5, Jiffy had concurrent usage

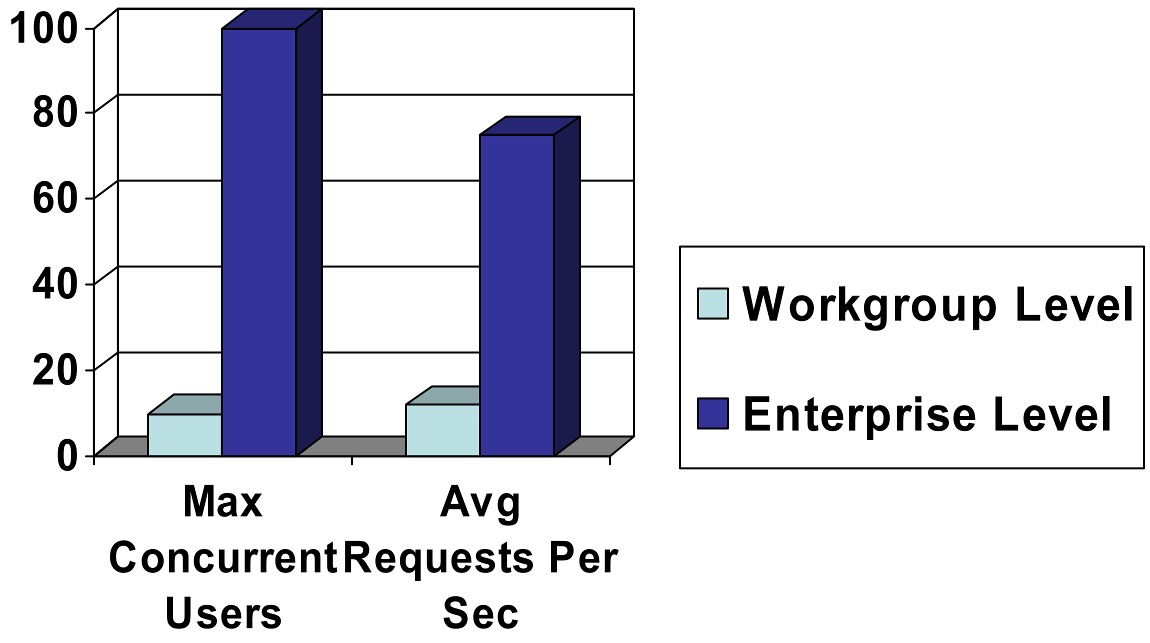


FIGURE 5. Original vs. Current Performance Comparison

problems that generally prevented (masked) response time from becoming an issue. That being said, a few portions of the application did suffer from non-concurrent usage related poor response times. Those areas were targeted for code rewrites and their response times improved accordingly.

Jiffy's main performance problems stem from the early development of the application in a workgroup environment without a guiding vision for the scalability required in an enterprise application. The Jiffy development team's job was to take the existing, very successful workgroup application and turn it into an enterprise level application in both functionality and performance. To that end they did many things correctly, made some mistakes, but ultimately set themselves up for successful implementation of future improvements to Jiffy.

Conversion of the Database

The first decision made to propel Jiffy to the enterprise level was to replace Microsoft Access with Oracle as the database. Since Access was not designed to be an enterprise level database it would seem that this move would be an instant-win situation. However, the team had a significant learning curve interfacing Microsoft's IIS Server with Oracle. The initial solution used Microsoft's generic Object DataBase Connectivity (ODBC) drivers to interface with Oracle, but response times turned out to be unacceptable. The better solution was to use Oracle's Oracle Objects 4 OLE (OO4O) drivers for the interface. Response times improved such that database calls are no longer considered to be an issue.

Once a satisfactory interface to Oracle was in place the team decided to move as much of the database interaction logic from application code into stored procedures that reside in the database. Several benefits were gained from this decision. First, the team was better able to consolidate the work the database had to do in one location. Since the application can accept automatic feeds directly into the database, this allowed it to perform its business logic no matter the source of the inputs. Secondly, since the data-related business logic was being executed in the database the application had access to native database routines for execution. The routines ran faster because of that tight coupling. But speed meant nothing if the application could not remain operational.

The Jiffy team went through a particularly rough phase early on in the conversion. Jiffy was prone to crash with very little logging to assist with debugging. A remedy that seemed to lower the chances of a crash was to have a weekly (or sometimes more frequent) reboot of the web server. For an application that was required to be online at all times this was unacceptable. In any case, the weekly reboots did not prevent crashes as desired. The Jiffy team sought the assistance of a Microsoft consultant in analyzing binary-level crash dumps and he was able to help determine the problem. A fix was quickly implemented and Jiffy has been crash-free ever since. Not all of the changes the team tried to eliminate the crashes were successful, but the lessons-learned were invaluable for later development.

Developing software is often a balance between delivering the desired functionality at all costs versus developing code that is easy to maintain for the developers. Sometimes those two goals go hand-in-hand. In the Jiffy application the development team made a decision to encapsulate many of the web-based pieces of database interface access logic into VBScript Classes. Those classes, in turn, would make the desired calls into the OO4O layer. Even though this decision made the developer's lives much easier, it caused a noticeable performance hit. Every one of the tables in the database has a corresponding VBScript class in Jiffy that is used to shuttle calls to it. Changing this is one of the main items on the list of future performance improvements.

All changes to web-based applications should be done in the context of a performance analysis. That is the only real way to know if the development was detrimental to the overall system. The VBScript classes were ported into COM-based compiled classes with robust logging capabilities. The port itself is not a major performance improvement. However, the porting was used as a way to perform database access profiling for Jiffy.

The Jiffy team established an automated test environment which exercised the entire system and generated a tremendous amount of COM Class logs. Developers used those logs to identify system performance bottlenecks caused by all too frequent round trips to the database. The Jiffy login sequence alone causes 54 database calls to execute for each user, every time. Three other actions combined with the login sequence account for about 70% of all database accesses. The Jiffy team has targeted those areas for rewrite as schedules allow. Performing this analysis means the team can focus development on four out of 117 data access pages and get tremendous benefit in the shortest amount of time (see Figure 6).

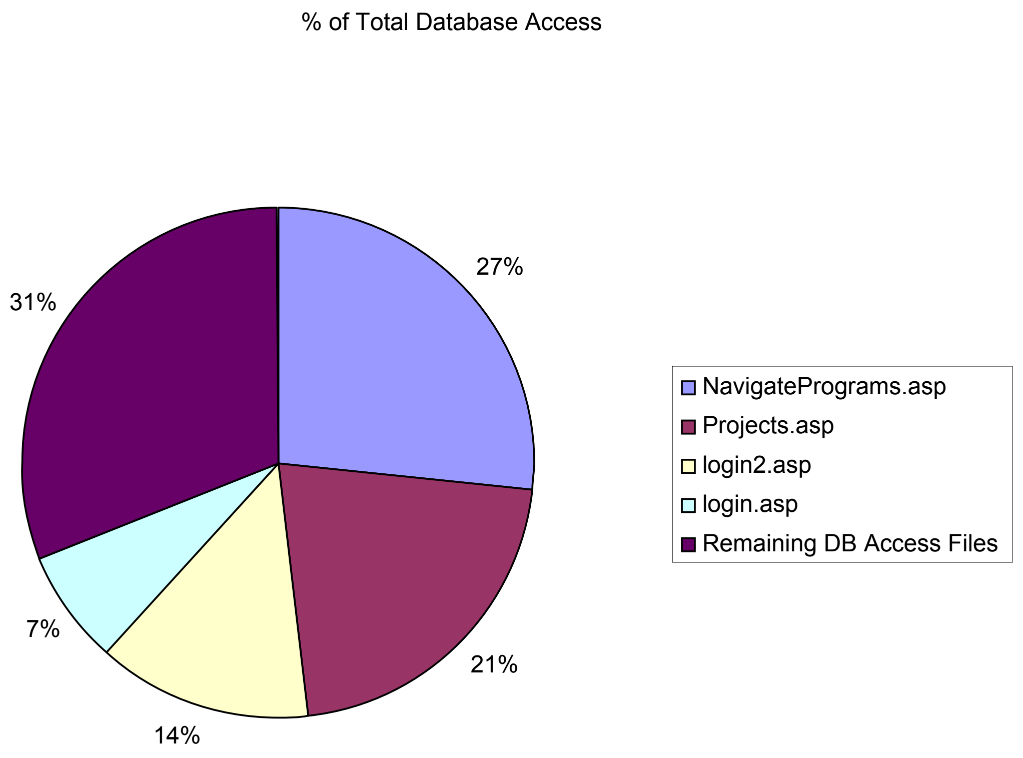


FIGURE 6. Percent of Total Database Access

Conclusion: Taking Jiffy from what was essentially a workgroup-level website to a scalable, enterprise-wide application required a tremendous effort from the entire product team. Occasionally decisions were made that turned out to be problematic, but the team continues to refine their analysis and development skills so that they make fewer and fewer mistakes. An important effort is to continue to automate the reliability and load testing so that potential changes can be quickly tested before committing to them.

All of the effort of providing a full-featured, scalable application would be wasted if the Jiffy team did not follow up with quality training and help desk support. All Jiffy users have the opportunity to attend application level hands-on training. For questions after that, they have the benefit of a multi-tiered customer support system. Jiffy is now positioned to have a long life at the enterprise level.

References:

1. "Challenges to Ensuring Secure .COM and .EDU Access to a Web-based Air Force Laboratory Program Management", Rico H., Hall F. et. al., Proceedings of the 2003 International Command and Control Research and Technology Symposium (ICCRTS), June 2003.

From Garage-Band to World Tour: Technical, Security, and Scalability Challenges of Migrating a Web-Based Program Management Tool from Workgroup-Level to Enterprise-Class in 24 Months

CCRTS - June 2004

Helen Rico & Fred Hall
Air Force Research Laboratory
Information Directorate
Rome, NY

Mike Maciolek
Northrop Grumman IT
Rome, NY

Presentation Outline

- Introduction
- Evolution of the Web-Based Architecture
- Security Considerations
- Application Scalability
- Conclusion

Information Directorate Background

- Headquartered in Rome, NY
- Formerly Rome Air Development Center, then Rome Laboratory, before becoming part of AFRL
- Mission:

  The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet warfighter needs.

- Our Business is Science

Introduction

- There was a need to:
  - Report information accurately and timely without retyping
  - Electronically create Laboratory Management Review forms
  - Have engineers and scientists return to R&D tasks in lieu of admin type duties
- The goal is to make reporting quick and easy for the Program Manager (scientists/engineers) and provide secure access to needed effort or program information.

Introduction (cont'd)

- Web-Based Program Management Tool "JIFFY"
- Accessible via any Web browser capable of 128-bit encryption
- Pulls Data from AF Standard Systems
- Accessible by non-.mil Domains
- Two and one-half year transition from Workgroup to Enterprise Level

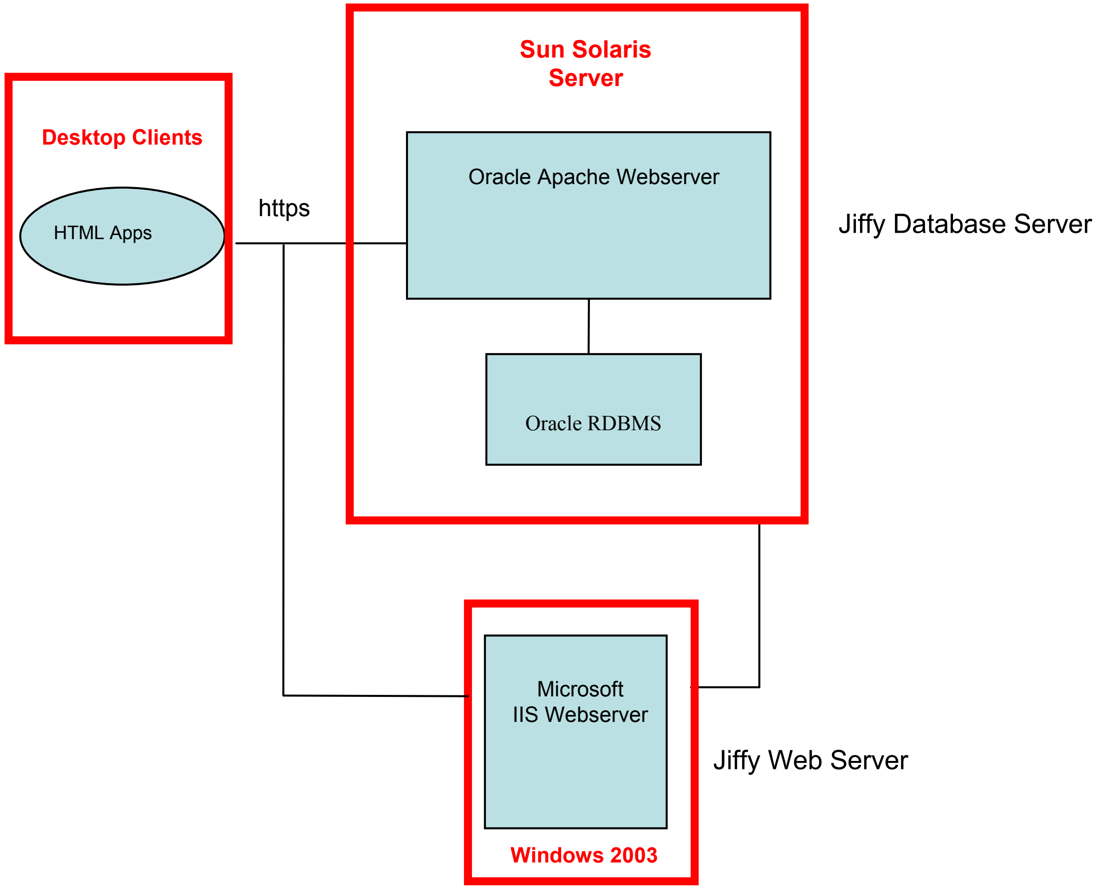
Presentation Outline

- Introduction
- Evolution of the Web-Based Architecture
- Security Considerations
- Application Scalability
- Conclusion

Architecture Basics

Two Main Pieces

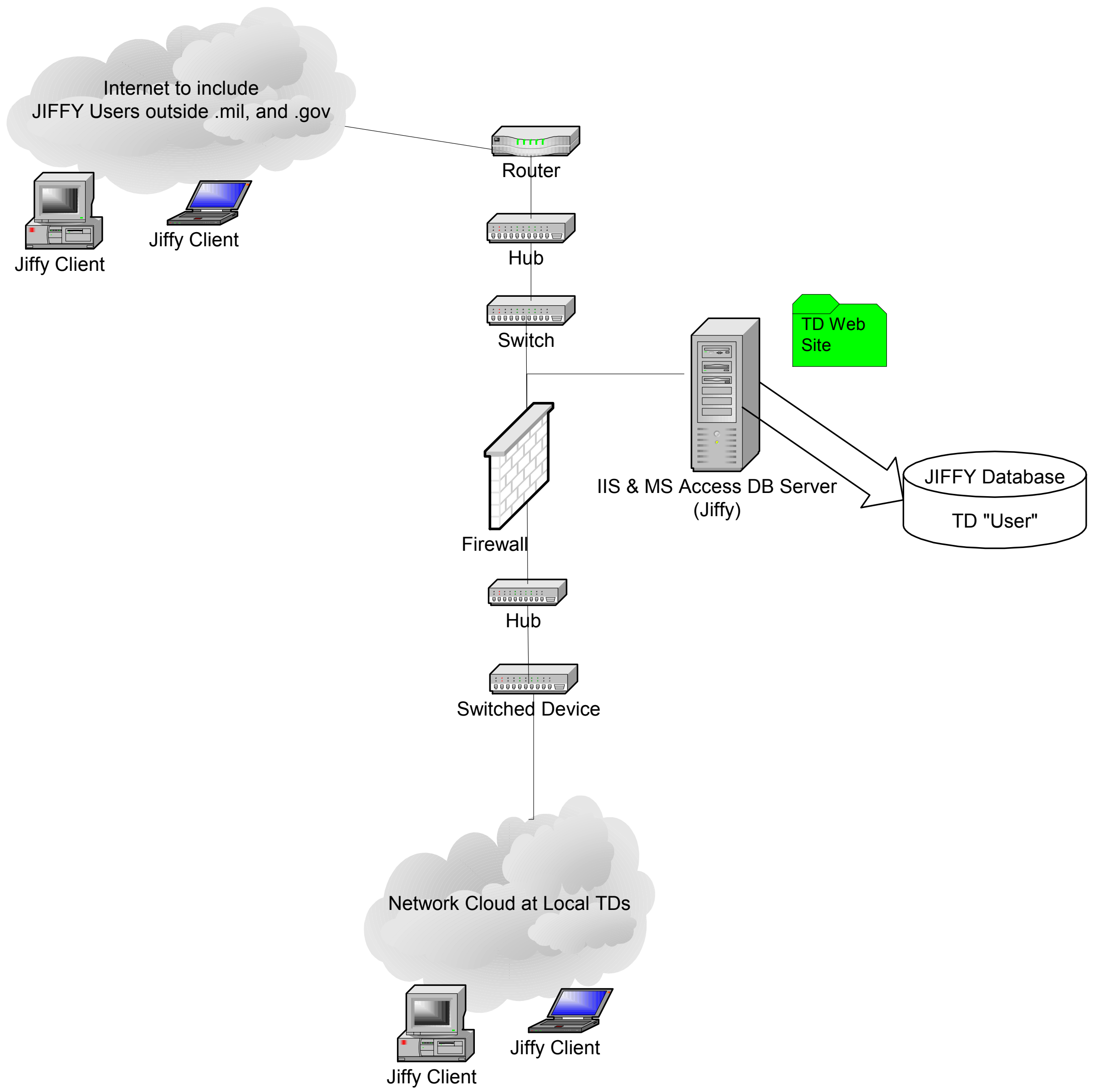
- Web Server
  - Handles user input and Graphical Display of information
- Database Server
  - Houses information gathered from AF Standard Systems and e-documents related to research programs



Workgroup Architecture

- Served a Handful of Users
- Development Staff: Two Part-Time Engineer/Programmers, One Full-Time Programmer
- Entirely Windows-Based
  - IIS Web Server
  - MS Access Database
  - Same Physical Computer
  - e-documents stored in Windows File System
- Outside Base Firewall to Facilitate .com/.edu Access

Workgroup Architecture (cont'd)



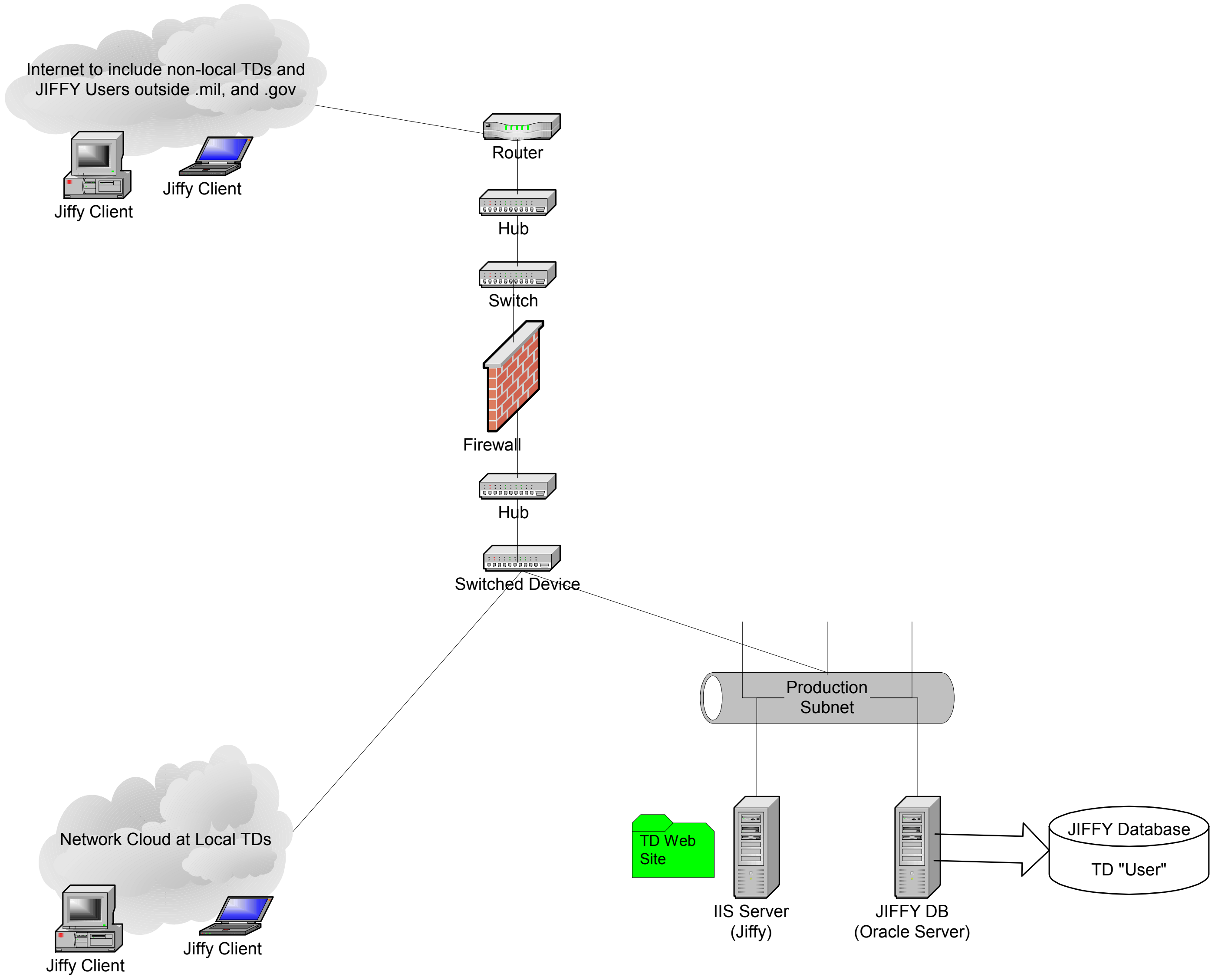
Workgroup Architecture (cont’d)

- Advantages
  - Good Performance
  - Low Maintenance Costs
  - Quick Development Cycle
- Disadvantages
  - Security Concerns
    - Computer Not Protected by Base Firewall
    - IIS, and MS Access vulnerabilities
    - Windows File System storage of e-documents
  - Not Very Scalable

Directorate-Wide Architecture

- Few Hundred Users
- Development Staff:
  - Six Programmers
  - Two Part-Time Program Managers
  - One Application Support Person
- Nine Month Development Timeframe
- Windows-Based IIS Web Server
  - Also used for storage of e-documents
- Sun Solaris Oracle Database Server
- Inside Base Firewall to Enhance Security
  - Firewall Rules Used to Facilitate .com/.edu Access

Directorate-Wide Architecture (cont'd)



Directorate-Wide Architecture (cont'd)

- **Advantages**
  - Higher Level of Security
  - More Robust and Scalable
  - Quick Development Cycle
- **Disadvantages**
  - Security Concerns
    - Windows File System storage of e-documents
  - Slight Performance Degradation

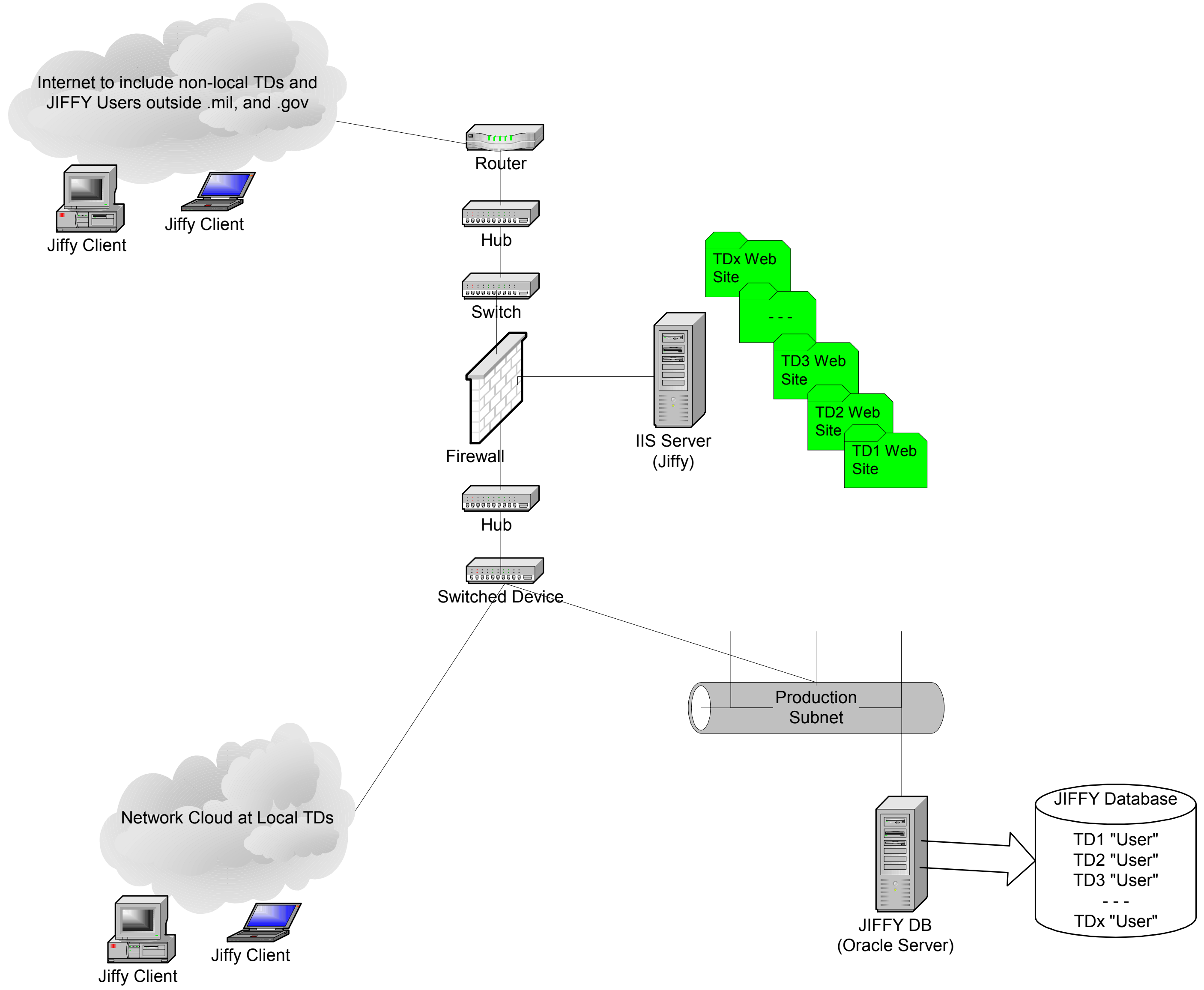
Enterprise-Wide Architecture

- Few Thousand Users Geographically-Dispersed across CONUS
- Development Staff:
  - Nine Programmers
  - Two S/W Testers
  - QA Person
  - Part-Time S/W Security Person
  - Program and Deputy Program Managers
  - Two Application Support People
  - Short-Term Paid Consultant
- 11 Month Development Timeframe

Enterprise-Wide Architecture (cont'd)

- **Windows-Based IIS Web Servers**
  - On separate Firewall "leg" (Extranet)
  - Three Physical Servers to Share Load
- **Sun Solaris Oracle Database Server**
  - e-documents stored in the database
- **Firewall Rules Used to Facilitate .com/.edu Access**

Enterprise-Wide Architecture (cont'd)



Enterprise-Wide Architecture (cont'd)

- Advantages
  - Higher Level of Security
  - More Robust and Scalable
  - Performance Improved

- Disadvantages
  - Longer Development Cycles
  - Higher Maintenance Requirements

Presentation Outline

- Introduction
- Evolution of the Web-Based Architecture
- Security Considerations
- Application Scalability
- Conclusion

Security Considerations

- Server Access
- Application Access
- User Roles and Permissions
- Traceability

Security Considerations (cont'd)

- Server Access
  - Contractor Access to Their Program Info is Crucial Feature
    - Requires .com/.edu Access to Server
  - DMZ (Extranet) Established to Facilitate Secure non-.mil Domain Access
  - Anti-Hacking Measures Incorporated Against; SQL Injection, Anonymous File System Access, Undesired Execute Privileges, URL Hijacking
  - e-documents Moved to Database Diminishes Exposure

Security Considerations (cont’d)

- Application Access
  - Trusted-Agent Account Nomination Process
  - Must Be US Citizen or I-551 “Green Card” Holder
- User Permissions
  - Role-Based Permissions
  - Row-Level Data Security

Security Considerations (cont'd)

- Traceability
  - Track User Activity in Critical Application Areas
  - Track Data Changes in Critical Application Areas
  - Web Server Logs Track User Activity Related to File Access
  - e-documents Moved to Database Diminishes Exposure
  - Allows Post-Mortem Analysis on Hacks
  - Assists in Debugging and Help-Desk Problem Resolution

Presentation Outline

- Introduction
- Evolution of the Web-Based Architecture
- Security Considerations
- Application Scalability
- Conclusion

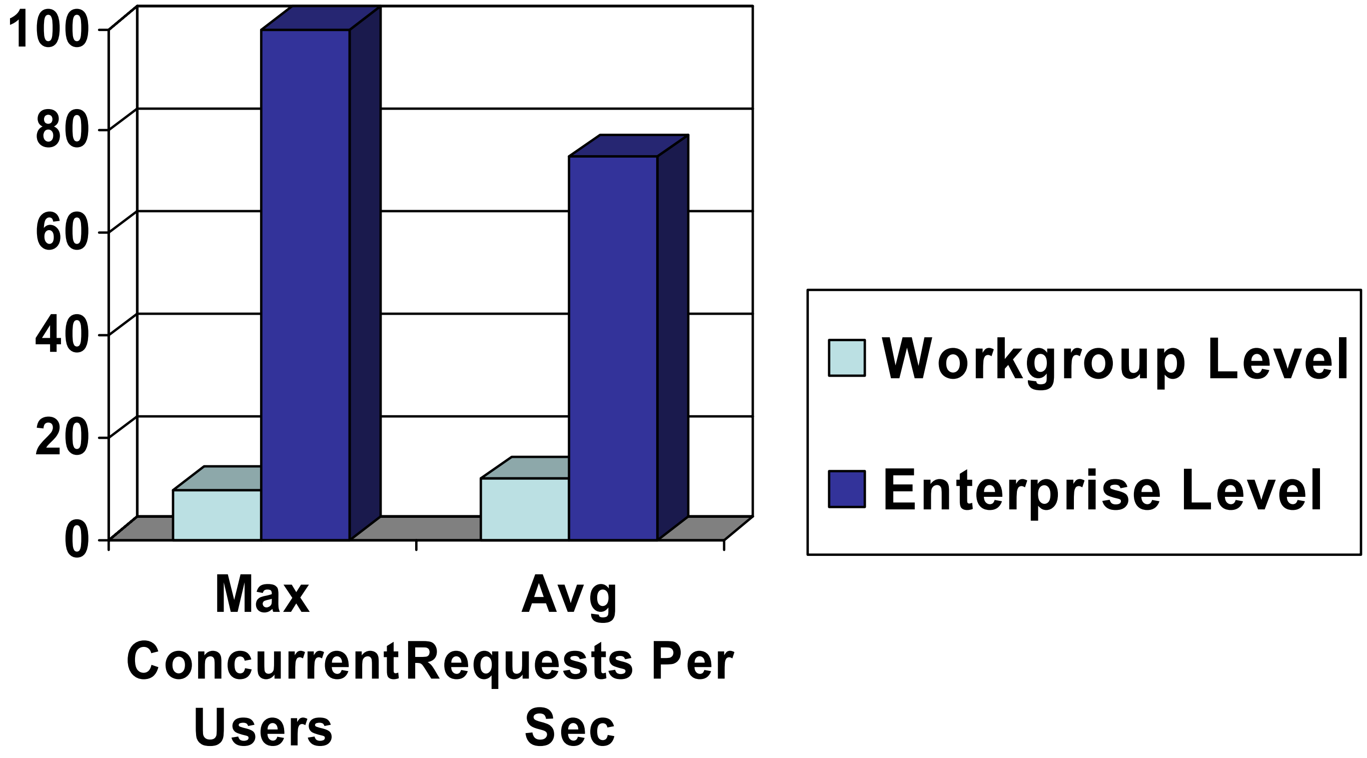
Application Scalability

- Performance Issues
- Database Conversion
- Design Testing
- Continuing Improvements

Application Scalability (cont'd)

- Application Performance
  - Response Time
    - Time for Web Server to Return Request
    - Average Number of Requests Served per Second
  - Concurrent Users
    - Number of Simultaneous Users that can Access a System
    - Normal Use Testing
    - Load Testing
  - Problem Areas
    - Early Development not Geared Toward Enterprise Scalability
    - Migration Time Constraints Led to Trade-offs

Application Scalability (cont'd)



Application Scalability (cont'd)

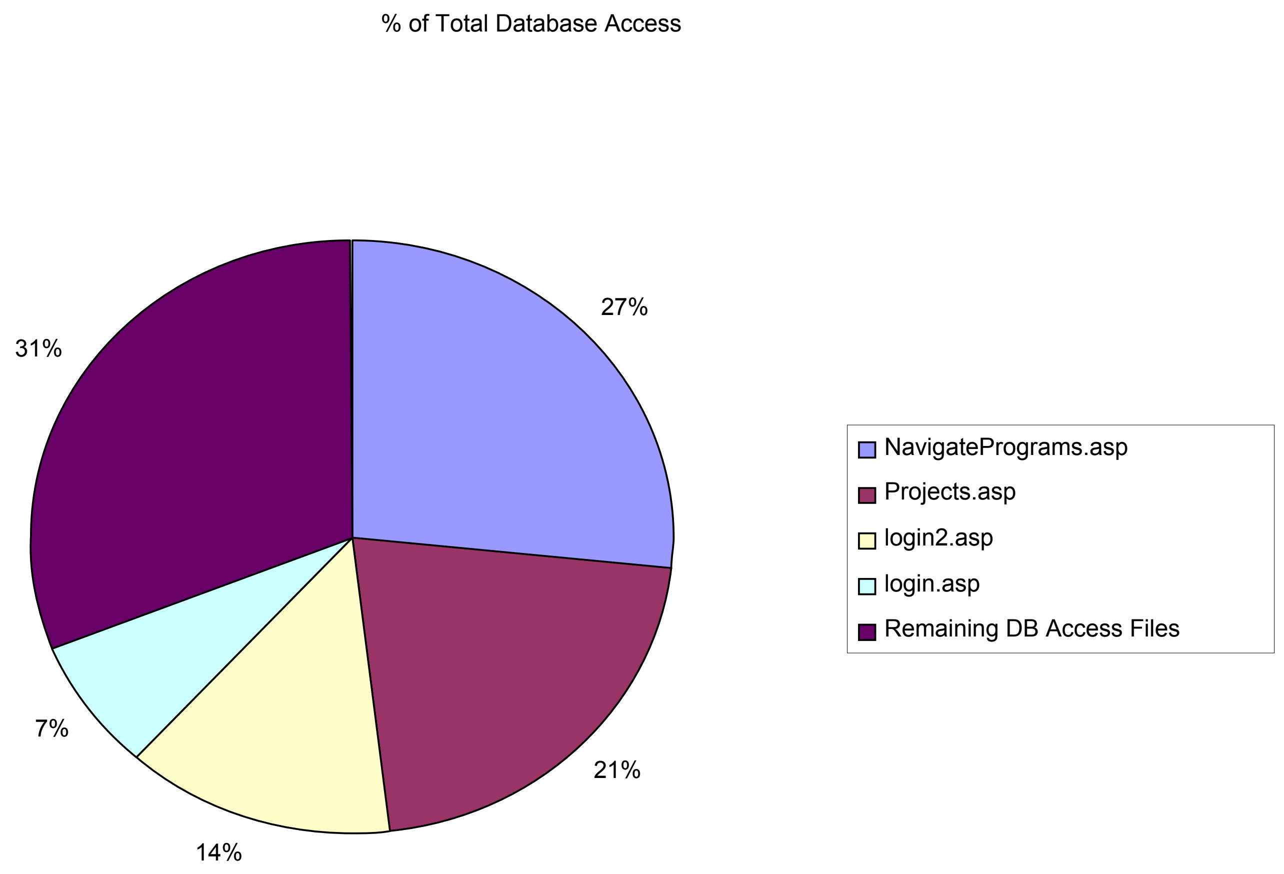
- Database Conversion
  - Interface Decisions
    - First Choice – MS Generic ODBC – poor performance
    - Moved to Oracle OO4O – significant performance gains
  - Stored Procedures
    - Moved Database Access Logic Into Stored Procedures
    - Consolidate Related Activities into APIs
      - Helps Developers
      - Allows Data Feeds to Properly Interact with System Logic
    - Use Native Database Routines for Speed and Functionality

Application Scalability (cont'd)

- Design Testing
  - VBScript Classes for Encapsulation
    - Early Development Decision to Encapsulate DB Access Logic in VBScript Classes
    - Poor Performance but Easily Maintainable
  - Profiling Components
    - Ported VBScript Classes to COM Classes
      - Compiled Executables
      - Superior Logging Capabilities
      - Helps Determine Data Access Bottlenecks
  - Automated Testing
    - Reliability and Regression Tests Developed
    - Load Tests Conducted for Performance Measurement

Application Scalability (cont'd)

Continuing Improvements



Application Scalability (cont'd)

- Continuing Improvements
  - Make Improvements Based on Application Profiling
  - Eye on Performance and Security

Conclusions

- Took Application from Workgroup to Enterprise in 24 Months
- Meets the Needs of the Diverse User Community
- Providing Help Desk and Hands-On Training is Crucial to Acceptance
- Well-Positioned for Long Life in the Enterprise

Comments/Questions?